AML Summary

Lukas Diebold

December 18, 2025

Contents

1 Math Preliminaries

Gibbs Distribution The Gibbs distribution is a probability distribution that assigns likelihoods to states based on a cost function, with lower-cost states being more probable. Given a set of states $x \in \mathcal{X}$, a cost function $E(x)$, and an inverse temperature parameter $\beta > 0$, the Gibbs distribution is

$$p(x) = \frac{1}{Z} e^{-\beta E(x)}$$

where the partition function Z ensures normalization

$$Z = \sum_{x \in X} e^{-\beta E(x)}$$

1.1 Conditional distribution of a multivariate Gaussian

Let

$$\begin{bmatrix} a \\ b \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix}, \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix} \right),$$

where Σ_{aa} and Σ_{bb} are covariance blocks and $\Sigma_{ab} = \Sigma_{ba}^\top$. Then the conditional distribution of a given b is again Gaussian with

$$\mathbb{E}[a \mid b] = \mu_a + \Sigma_{ab}\, \Sigma_{bb}^{-1}(b - \mu_b), \quad \mathrm{Cov}(a \mid b) = \Sigma_{aa} - \Sigma_{ab}\, \Sigma_{bb}^{-1}\, \Sigma_{ba}.$$

This identity underlies Gaussian process prediction, Bayesian linear regression posteriors, and Kalman filtering.

Derivation. Write the joint as a block Gaussian and use the Schur complement. The joint log-density (up to constants) is

$$\ell(a, b) = \tfrac{1}{2} \begin{bmatrix} a - \mu_a \\ b - \mu_b \end{bmatrix}^\top \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}^{-1} \begin{bmatrix} a - \mu_a \\ b - \mu_b \end{bmatrix}.$$

Completing the square in a (or applying the standard conditional Gaussian formula) yields

$$\mathbb{E}[a \mid b] = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(b - \mu_b), \quad \mathrm{Cov}(a \mid b) = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}.$$

Equivalently, these follow from the block inversion identity and the Schur complement of Σ_{bb}.

1.2 Schur complement

Note. For a block matrix $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ with D invertible, the **Schur complement of D in M** is

$$S = A - BD^{-1}C.$$

Key identities (when the required inverses exist):

- Determinant: $\det(M) = \det(D)\, \det(S)$.
- Block inverse: $M^{-1} = \begin{bmatrix} S^{-1} & -S^{-1}BD^{-1} \\ -D^{-1}CS^{-1} & D^{-1} + D^{-1}CS^{-1}BD^{-1} \end{bmatrix}$.
- If M is symmetric positive (semi)definite and D is invertible, then S is also positive (semi)definite.

Symmetrically, if A is invertible, the Schur complement of A is $D - CA^{-1}B$. In Gaussian conditioning, $\mathrm{Cov}(a \mid b)$ equals the Schur complement of Σ_{bb} in the joint covariance.

2 Conceptual Foundation

2.1 What is Machine Learning?

"ML is a mathematization of epistemolagy!". In philosophy, it is the science of knowledge, the science of what can be known. This is relevant, because in ML we are interested in systems that produce/generate knowledge.

The goal is then to observe 'reality' and draw conclusions from the observations. This can be seen as a perception-action cycle. Where perception is the result of our observations (typically in a data space $\mathcal{X}$) and the actions are part of a hypothesis space. To go from the data space to the hypothesis class $\mathcal{C}$ we generally use an algorithm A. See Figure 1 for an overview.

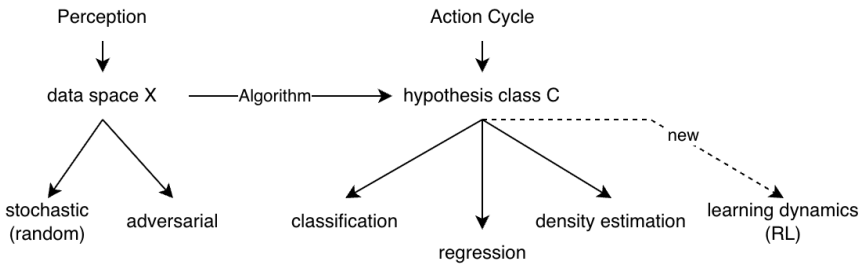

Figure 1: Overview ML

Information processing occurs when $|\mathcal{X}| \gg |\mathcal{C}|$. Taking the example of combinatorial optimization problems, $|\mathcal{X}|$ would be the space of weighted graphs, and we look for a color of the graph or something similar. Then $|\mathcal{X}| \approx K^{\binom{n}{2}}$ and $|\mathcal{C}| \approx e^{n \log n}$ so we obserge that $|\mathcal{X}|$ is much larger.

2.2 Conceptual foundation of inference

1. Perception of realits is mediated by data of senses/sensors
2. Data are stochastic $\rightarrow$ porbabilistic
3. Sensing restricts us to selected aspects of reality
4. Humans interpret data by a huge reduction in degrees of freedom
   $|x| \gg |e|$ (space of graphs $\gg$ space of colorings, cycles, spanning trees)
5. Tuple ({data }, {hypotheses }) define models:

$$\begin{aligned} \mathcal{A} : \mathcal{X} &\longrightarrow \mathcal{C} \\ x &\longmapsto c = \mathcal{A}(x) \end{aligned}$$

6. Experiments ϵ provide us with data
7. Learning means interpreting X w.r.t hypotheses $\mathcal{C}$

2.3 Artificial Intelligence

Taking a high level view. We live in very high dimensional data, which we are not able to fully process. We use algorithms to make sense of the data, and this then informs our values, from this we get the following relation

$$\text{Data} \longrightarrow \text{ Algorithms } \mathcal{A} \longrightarrow \text{ Values}$$

In epistemology, we differentiate between **deduction** and **induction**. Deduction is a form of reasoning in which the conclusion follows necessarily from the premises, while induction tries to generalize, that is, the conclusion goes beyond the premises and generally probabilistic. We can construct a model in which deduction and induction form a **feedback loop**, not two isolated

methods. In simpler terms, on one side we try to formulate axioms from our observations (empirical data), and on the other side we then use these axioms and derive logical consequences from them. This is not anything new, and this cycle generally informs the model of "Theory, Experiment, Computation" in science. What has changed with ML/AI is that we're now in the era of non-parametric modeling.

2.4 Extracting Value from Data - What is the problem?

- Algorithms that process inputs with noise compute random variables as outputs!
- Algorithms should compute typical solutions!
- When do algorithms generalize over noise/model mismatch?
- How can algorithms autonomously improve performance?

2.5 What does Generalization mean?

- Out-of-sample risk

$$\theta^{\star}\left(X'\right) \sim \mathbb{P}^{\mathcal{A}}(\theta \mid X) \in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{\theta|X'}\mathbb{E}_{X''|X'}R\left(\theta, X''\right)$$

  where X' is the training data and X'' is the test data, so thiw is the risk where θ is conditioned on the training data (trained the model).

- Log loss of posterior (risks and probabilities are dependent!)

$$\begin{aligned}\theta^{\star}\left(X'\right) \sim \mathbb{P}^{\mathcal{A}}(\theta \mid X) &\in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{\theta|X'}\mathbb{E}_{X''|X'}\left(-\log \mathbb{P}\left(\theta \mid X''\right)\right) \\ &\in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{\theta|X'}\mathbb{E}_{X''|X'}\left(\beta R\left(\theta, X''\right) + \log Z\right)\end{aligned}$$

- Posterior agreement

$$\theta^{\star}\left(X'\right) \sim \mathbb{P}^{\mathcal{A}}(\theta \mid X) \in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{X''|X'}\left(-\log \mathbb{E}_{\theta|X'}\mathbb{P}\left(\theta \mid X''\right)\right)$$

2.6 Conceptional Foundation of Inference

- Our perception of "our world" / reality is mediated by data of senses / sensors.
- Our data are influenced by chance.
- Creatures interpret selected aspects of reality by hypotheses to "survive and reproduce"
- Data and hypotheses define models to enable judgements, decisions and actions.
- AI / ML: Algorithms define relations of data and hypotheses, e.g., they select models !

3 Fundamentals of Machine Learning

Machine learning is fundamentally about inferring models from data. At its core lies Bayes' rule, which relates the posterior distribution (model given data) to the likelihood and prior:

$$\mathbb{P}(\text{model} \mid \text{data}) = \frac{\mathbb{P}(\text{ data } \mid \text{ model })\mathbb{P}(\text{ model })}{\mathbb{P}(\text{ data })}$$

In the **maximum likelihood (ML)** approach, we select the model that maximizes the likelihood of observing the data:

$$\widehat{\text{model}} \in \arg\max_{\text{model}} \mathbb{P}(\text{data} \mid \text{model})$$

Under regularity conditions, the ML estimator $\widehat{\text{model}}_n$ is consistent, asymptotically normal, and asymptotically efficient.

Consistency: A point estimator $\hat{\theta}_n$ of the parameter $\theta = \theta_0$ is consistent if it converges in probability to the true parameter:

$$\forall \varepsilon > 0, \mathbb{P}\left(\left|\hat{\theta}_n - \theta_0\right| > \varepsilon\right) \xrightarrow{n\to\infty} 0$$

More formally, using the ε-δ definition:

$$\forall \theta, \forall \varepsilon, \delta > 0, \exists n_0, \forall n > n_0, \mathbb{P}\left(\left|\widehat{\theta}_n - \theta\right| < \varepsilon\right) > 1 - \delta$$

Efficiency: An estimator $\hat{\theta}_n$ is efficient if it achieves the minimum mean squared error among all estimators:

$$\hat{\theta}_n = \arg\min_{\hat{\theta}} \mathbb{E}\left[\left(\hat{\theta}_n - \theta_0\right)^2\right]$$

The question then arises: how precisely can we estimate θ given n samples? The Cramér-Rao bound provides a fundamental lower bound on the variance of any unbiased estimator.

3.1 Efficiency: Cramér-Rao Bound

Problem: What is the best achievable precision for parameter estimation?

Given a likelihood $p(y \mid \theta)$ for $\theta \in \Theta$ and data $y_1, \ldots, y_n \sim p(y \mid \theta = \theta_0)$, we ask: How precisely can we estimate $\theta = \theta_0$ given n samples?

For an estimator $\hat{\theta}(y_1, \ldots, y_n)$, we measure precision via the mean squared error:

$$\mathbb{E}_{y|\theta}\left[(\hat{\theta} - \theta)^2\right]$$

Key definitions:

- Score: $\Lambda = \frac{\partial}{\partial\theta} \log p(y \mid \theta) = \frac{\frac{\partial}{\partial\theta} p(y|\theta)}{p(y|\theta)}$
- Bias: $b_{\hat{\theta}} = \mathbb{E}_{y|\theta}\left[\hat{\theta}(y_1, \ldots, y_n)\right] - \theta$

Expected score: The score has zero mean.

$$\begin{aligned}\mathbb{E}_{y|\theta}[\Lambda] &= \int p(y \mid \theta) \frac{\frac{\partial}{\partial\theta} p(y \mid \theta)}{p(y \mid \theta)} dy \\ &= \frac{\partial}{\partial\theta} \int p(y \mid \theta) dy = \frac{\partial}{\partial\theta} 1 = 0\end{aligned}$$

Score-estimator product:

$$\begin{aligned}\mathbb{E}_{y|\theta}[\Lambda\hat{\theta}] &= \int p(y \mid \theta) \frac{\frac{\partial}{\partial\theta} p(y \mid \theta)}{p(y \mid \theta)} \hat{\theta} dy \\ &= \frac{\partial}{\partial\theta}\left(\int p(y \mid \theta)\hat{\theta} dy\right) \\ &= \frac{\partial}{\partial\theta}\left(\mathbb{E}_{y|\theta}\hat{\theta}\right) = \frac{\partial}{\partial\theta}\left(b_{\hat{\theta}} + \theta\right) = \frac{\partial}{\partial\theta} b_{\hat{\theta}} + 1\end{aligned}$$

Cross-correlation:

$$\mathbb{E}_{y|\theta}\left[(\Lambda - \mathbb{E}\Lambda)\left(\hat{\theta} - \mathbb{E}\hat{\theta}\right)\right] = \mathbb{E}_{y|\theta}[\Lambda\hat{\theta}] - \mathbb{E}_{y|\theta}[\Lambda]\mathbb{E}\hat{\theta} = \mathbb{E}_{y|\theta}[\Lambda\hat{\theta}]$$

since $\mathbb{E}[\Lambda] = 0$.

Cauchy-Schwarz inequality: Applying Cauchy-Schwarz to the cross-correlation:

$$\left(\mathbb{E}_{y|\theta}[\Lambda(\hat{\theta} - \mathbb{E}\hat{\theta})]\right)^2 \leq \mathbb{E}_{y|\theta}\left[\Lambda^2\right]\mathbb{E}_{y|\theta}\left[(\hat{\theta} - \mathbb{E}\hat{\theta})^2\right]$$

Expanding the right-hand side:

$$\begin{aligned}\mathbb{E}_{y|\theta}\left[(\hat{\theta} - \mathbb{E}\hat{\theta})^2\right] &= \mathbb{E}_{y|\theta}\left[(\hat{\theta} - \theta + \theta - \mathbb{E}\hat{\theta})^2\right] \\ &= \mathbb{E}_{y|\theta}\left[(\hat{\theta} - \theta)^2\right] - b_{\hat{\theta}}^2\end{aligned}$$

Therefore:

$$\left(\frac{\partial}{\partial\theta}b_{\hat{\theta}} + 1\right)^2 \leq \mathbb{E}_{y|\theta}\left[\Lambda^2\right]\left(\mathbb{E}_{y|\theta}\left[(\hat{\theta} - \theta)^2\right] - b_{\hat{\theta}}^2\right)$$

Rearranging yields the **Cramér-Rao bound**:

$$\mathbb{E}_{y|\theta}\left[(\hat{\theta} - \theta)^2\right] \geq \frac{\left(\frac{\partial}{\partial\theta}b_{\hat{\theta}} + 1\right)^2}{\mathbb{E}_{y|\theta}\left[\Lambda^2\right]} + b_{\hat{\theta}}^2$$

Fisher information: The expected squared score is called the Fisher information:

$$I(\theta) := \mathbb{E}_{y|\theta}\left[\Lambda^2\right] = \int p(y \mid \theta)\left(\frac{\partial}{\partial\theta}\log p(y \mid \theta)\right)^2 dy$$

It measures how much information the data contains about the parameter θ. Higher Fisher information means we can estimate θ more precisely.

Remarks:

- For unbiased estimators ($b_{\hat{\theta}} = 0$), the bound simplifies to $\mathbb{E}[(\hat{\theta} - \theta)^2] \geq 1/I(\theta)$.
- The bound reveals a trade-off for biased estimators: reducing bias derivative $\frac{\partial}{\partial\theta}b_{\hat{\theta}}$ vs. reducing squared bias $b_{\hat{\theta}}^2$. Unbiased estimators are not always optimal!

Fisher information for n i.i.d. samples:

$$\begin{aligned}I^{(n)}(\theta) &= \mathbb{E}_{y_1,\ldots,y_n|\theta}\left[\Lambda^2\right] \\ &= \mathbb{E}\left[\left(\frac{\partial}{\partial\theta}\log p\left(y_1,\ldots,y_n \mid \theta\right)\right)^2\right] \\ &= \mathbb{E}\left[\left(\sum_{i=1}^{n}\frac{\partial}{\partial\theta}\log p\left(y_i \mid \theta\right)\right)^2\right] = \mathbb{E}\left[\left(\sum_{i=1}^{n}\Lambda_i\right)^2\right] \\ &= \sum_{i=1}^{n}\mathbb{E}\left[\Lambda_i^2\right] + \sum_{i\neq j}\mathbb{E}\left[\Lambda_i\right]\mathbb{E}\left[\Lambda_j\right] \\ &= \sum_{i=1}^{n}\mathbb{E}\left[\Lambda_i^2\right] = nI(\theta)\end{aligned}$$

where the cross-terms vanish because $\mathbb{E}[\Lambda_i] = 0$ and the samples are independent.

Key insight: The Fisher information of n i.i.d. random variables is n times the Fisher information of a single random variable. This shows that precision improves linearly with sample size.

4 Regression

4.1 Act 1: High-dimensional regression is unstable

We assume X and y are distributed according to a distribution p_* (i.e. $X, y \sim p_*$), where the output follows a noisy linear model:

$$y = f_*(x) + \varepsilon \quad \text{with } \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

Here f_* is the true (unknown) regression function and ε is additive Gaussian noise with variance σ^2. Our task is to estimate f_* from training data $D = \{(x_i, y_i)\}_{i=1}^n \sim p_*$.

The problem in this form is not tractable because the space of all possible functions is too large. We therefore restrict ourselves to linear functions:

$$f_*(x) = \beta^\top x$$

where $\beta \in \mathbb{R}^d$ is a parameter vector. Given the Gaussian noise assumption, each observation has likelihood $p(y_i|x_i, \beta) = \mathcal{N}(\beta^\top x_i, \sigma^2)$. We solve for β using Maximum Likelihood Estimation (MLE):

$$\begin{aligned}
\widehat{\beta} &= \arg\max_{\beta \in \mathbb{R}^d} p(D \mid \beta) \\
&= \arg\max_{\beta} \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \beta^\top x_i)^2}{2\sigma^2}\right) \\
&= \arg\min_{\beta} \sum_{i=1}^n (y_i - \beta^\top x_i)^2 \\
&= \arg\min_{\beta} \mathrm{MSE}(D, \beta)
\end{aligned}$$

Maximizing the log-likelihood is equivalent to minimizing the mean squared error (MSE). The closed-form solution depends on whether we have more features than samples or vice versa:

$$\begin{aligned}
\widehat{\beta} &= (X^\top X)^{-1} X^\top y \quad \text{(when } d < n \text{, more samples than features)} \\
&= X^\top (X X^\top)^{-1} y \quad \text{(when } d > n \text{, more features than samples)}
\end{aligned}$$

These are algebraically equivalent by the Woodbury matrix identity. The first formula is the standard *ordinary least squares (OLS)* estimator, where

$$X = \begin{bmatrix} -x_1- \\ \vdots \\ -x_n- \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

This estimator has some interesting properties. It is unbiased and, by the Gauss-Markov Theorem, it is the best linear unbiased estimator (BLUE), i.e. it attains the smallest variance among all linear unbiased estimators. Thus, from the formula

$$\text{error} = \text{bias}^2 + \text{variance} + \text{noise}$$

we find that this estimator is the one with the smallest error of all the unbiased estimators. Then why does no-one use this estimator? If we introduce a bit of bias, we can significantly reduce the variance.

To understand the instability, we analyze $\mathrm{Var}(\hat{\beta})$ using the singular value decomposition (SVD) $X = UDV^\top$, where $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{d \times d}$ are orthogonal, and D is diagonal with singular values $D_{11} \geq D_{22} \geq \ldots \geq 0$. Plugging this into the OLS formula:

$$\hat{\beta} = (X^\top X)^{-1} X^\top y = (VD^\top U^\top U D V^\top)^{-1} V D^\top U^\top y = V D^{-1} U^\top y$$

Since $y = X\beta_* + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$, and we multiply y by the deterministic matrix $VD^{-1}U^\top$, the estimator $\hat{\beta}$ is also Gaussian. Its variance is:

$$\mathrm{Var}(\hat{\beta}) = \mathrm{Var}(VD^{-1}U^\top y) = VD^{-1}U^\top \,\mathrm{Var}(y) UD^{-1}V^\top = \sigma^2 VD^{-2}V^\top = \sigma^2 \sum_{i \leq r} \frac{1}{D_{ii}^2} V_i V_i^\top$$

where $r = \text{rank}(X)$ and V_i is the i-th column of V (the i-th right singular vector).

The problem: In high-dimensional data, features are often correlated (e.g., pixel intensities in images, gene expressions). This makes X close to low-rank, so several singular values D_{ii} are very small. The variance contributions $1/D_{ii}^2$ then explode for these directions, causing massive instability in $\hat{\beta}$ even though it remains unbiased. Small noise in y gets amplified enormously in directions with small singular values, leading to wild predictions on test data.

4.2 Act 2: Stability via Regularization

The solution to the variance blow-up is to introduce regularization, which adds a small amount of bias in exchange for a large reduction in variance. We can derive regularization naturally from a Bayesian perspective.

The typical process of **Bayesian inference** goes through the following stages:

1. Prior $\beta \sim \mathcal{N}(0, \tau^2 I)$ — we assume β is drawn from a Gaussian centered at zero with variance τ^2. This encodes our belief that coefficients should not be too large.
2. Likelihood $p(D|\beta) = \prod_i \mathcal{N}(y_i|\beta^\top x_i, \sigma^2)$ — same Gaussian noise model as before.
3. Posterior (via Bayes' rule) $p(\beta|D) \propto p(\beta)p(D|\beta) \propto \exp\left(-\frac{1}{2\sigma^2}\text{MSE}(D,\beta) - \frac{1}{2\tau^2}\|\beta\|^2\right)$

The posterior combines the likelihood (fit to data) with the prior (regularization). To derive the optimization objective, we use the fact that maximizing the posterior probability is equivalent to minimizing its negative logarithm. From Bayes' rule:

$$p(\beta|D) \propto p(\beta)p(D|\beta) = \mathcal{N}(0, \tau^2 I) \cdot \prod_i \mathcal{N}(y_i|\beta^\top x_i, \sigma^2)$$

Taking the negative log (up to constant terms):

$$-\log p(\beta|D) \propto \underbrace{-\log p(\beta)}_{-\frac{1}{2\tau^2}\|\beta\|^2 + \text{const}} + \underbrace{-\log p(D|\beta)}_{-\sum_i \log \mathcal{N}(y_i|\beta^\top x_i, \sigma^2)}$$

For Gaussian distributions, $-\log \mathcal{N}(y|\mu, \sigma^2) = \frac{(y-\mu)^2}{2\sigma^2} + \text{const}$, so:

$$-\log p(\beta|D) \propto \frac{1}{2\tau^2}\|\beta\|^2 + \frac{1}{2\sigma^2}\sum_i (y_i - \beta^\top x_i)^2$$

Minimizing this gives the MAP (maximum a posteriori) estimate:

$$\begin{aligned}\hat{\beta}_{\text{MAP}} &= \arg\min_\beta \left[\frac{1}{2\sigma^2}\sum_i (y_i - \beta^\top x_i)^2 + \frac{1}{2\tau^2}\|\beta\|^2\right] \\ &= \arg\min_\beta \left[\sum_i \left(y_i - \beta^\top x_i\right)^2 + \lambda\|\beta\|^2\right]\end{aligned}$$

This is precisely **ridge regression** with regularization parameter $\lambda = \sigma^2/\tau^2$. The ℓ^2 penalty $\|\beta\|^2$ shrinks coefficients toward zero. If we instead use a Laplace prior $p(\beta) \propto \exp(-|\beta|/\tau)$ with heavier tails, we obtain **lasso regression** with an ℓ^1 penalty $\|\beta\|_1$, which promotes sparsity.

The prior variance τ^2 controls the bias-variance trade-off: small τ^2 (big λ) means strong regularization (more bias, less variance), while large τ^2 (small λ) recovers OLS (no bias, high variance). The MAP (maximum a posteriori) solution is:

$$\hat{\beta}_{\text{MAP}} = \left(X^\top X + \lambda I\right)^{-1} X^\top y \tag{1}$$

Derivation. Start from the MAP/ridge objective in matrix form

$$J(\beta) = \sum_i (y_i - x_i^\top \beta)^2 + \lambda\,\|\beta\|^2 \;=\; \|y - X\beta\|^2 + \lambda\,\beta^\top\beta.$$

Differentiate and set the gradient to zero:

$$\begin{aligned}\nabla_\beta J(\beta) &= -2X^\top(y - X\beta) + 2\lambda\,\beta \\ &= 2\,(X^\top X + \lambda I)\beta - 2X^\top y \;=\; 0.\end{aligned}$$

Thus the normal equations are $(X^\top X + \lambda I)\,\hat{\beta} = X^\top y$. For $\lambda > 0$, the matrix $X^\top X + \lambda I$ is positive definite and hence invertible, which yields the closed form in (1).

Compare this to OLS: the regularization term λI is added to $X^\top X$ before inversion, preventing ill-conditioning. Using SVD again to analyze the variance:

$$\mathrm{Var}(\hat{\beta}_{\mathrm{MAP}}) = \sigma^2 \sum_{i \le r} \frac{D_{ii}^2}{\left(D_{ii}^2 + \lambda\right)^2} V_i V_i^\top \tag{2}$$

The key is the **shrinkage factor** $\frac{D_{ii}^2}{(D_{ii}^2+\sigma^2/\tau^2)^2}$. For large singular values ($D_{ii}^2 \gg \sigma^2/\tau^2$), this is close to $1/D_{ii}^2$ (like OLS). For small singular values ($D_{ii}^2 \ll \sigma^2/\tau^2$), the factor is approximately $\tau^4 D_{ii}^2/\sigma^4$, which decays much more slowly than $1/D_{ii}^2$. This prevents variance blow-up in the problematic low-variance directions, stabilizing the estimator at the cost of introducing bias (shrinking coefficients toward zero).

4.3 Act 3: Polynomial regression via kernels

Now we change our assumption for $f_*(x)$ to allow for nonlinear functions. We model f_* as a linear function in an infinite-dimensional feature space:

$$f_*(x) = \varphi(x)^\top \beta_*$$

where $\beta_* \in \mathbb{R}^\infty$ and $\varphi(x)$ maps each input to an infinite-dimensional polynomial feature representation:

$$\varphi(X) = K_x \left(\frac{x_1^{\alpha_1} \ldots x_d^{\alpha_d}}{\sqrt{\alpha_1! \ldots \alpha_d!}} \right)_{\alpha \in \mathbb{N}^d}$$

This includes all polynomial terms of all degrees. The normalization by factorials ensures the inner product has a clean closed form.

Remarkably, the inner product of two infinite-dimensional feature vectors yields the radial basis function (RBF) kernel. For $x, x' \in \mathbb{R}^a$:

$$\begin{aligned}\varphi(x)^\top \varphi\left(x'\right) &= K_{RBF}\left(x, x'\right) \\ &= \exp\left(-\frac{1}{2}||x - x'||^2\right)\end{aligned}$$

This follows from the Taylor expansion of the exponential function. The RBF kernel measures similarity: it is 1 when $x = x'$ and decays as points move apart.

We still want to minimize the MSE, but now in the infinite-dimensional feature space:

$$\begin{aligned}\hat{\beta} &= \arg\min_{\beta \in \mathbb{R}^\infty} \frac{1}{n} \sum_{i \le n} \left(y_i - \varphi\left(x_i\right)^\top \beta\right)^2 \\ &= \Phi^\top \left(\Phi\Phi^\top\right)^{-1} y\end{aligned}$$

where

$$\Phi = \begin{bmatrix} \varphi(x)^\top \\ \varphi\left(x_2\right)^\top \\ \varphi\left(x_n\right)^\top \end{bmatrix} \in \mathbb{R}^{n \times \infty}$$

Despite β living in infinite dimensions, the representer theorem guarantees the solution lies in the span of the training features, so we can work with the $n \times n$ Gram matrix $\Phi\Phi^\top$ instead of the infinite-dimensional feature space directly.

To make a prediction at test point x_*, we compute:

$$\begin{aligned}\hat{y}_* &= \varphi(x_*)^\top \hat{\beta} \\ &= \varphi(x_*)^\top \Phi^\top \left(\Phi\Phi^\top\right)^{-1} y \\ &= k(x_*)^\top K^{-1} y\end{aligned}$$

where $k(x_*) = \left(\varphi(x_*)^\top \varphi(x_i)\right)_{1\le i\le n} = (K_{RBF}(x_*, x_i))_{1\le i\le n}$ is an n-dimensional vector of kernel evaluations between the test point and each training point, and $K_{ij} = \varphi(x_i)^\top \varphi(x_j) = K_{RBF}(x_i, x_j)$ is the $n \times n$ kernel matrix.

This is the **kernel trick**: we never explicitly construct the infinite-dimensional $\varphi(\cdot)$. Instead, we only compute inner products via the kernel function K_{RBF}, which can be evaluated in closed form. The prediction is a weighted combination of training outputs, where the weights depend on how similar the test point is to each training point.

The problem is that the inversion of the matrix is $O(n^3)$, which becomes costly for large datasets even though we avoided the infinite feature map explicitly.

4.4 Act 4: Neural Networks

We assume f_* has only a single, very wide hidden layer.

$$f_*(X) = \frac{1}{\sqrt{m}} \sum_{i\le m} \alpha_i \phi\left(\omega_i^\top X\right)$$

where ϕ is a nonlinear activation function (e.g. ReLU, tanh), and the network has m hidden units. The parameters are $\theta = \{\alpha_i, w_i\}_{i\le m}$, i.e. both the output weights α_i and the input weights w_i are learned. We initialize with

$$\theta_0 \sim \mathcal{N}\left(0, w^2\right)$$

and we update our parameters using gradient descent.

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla_\theta \text{ MSE}(D, \theta_t)$$

The gradient can be written in matrix form as

$$\nabla_\theta \text{MSE}(D, \theta_t) = \widetilde{\Phi}_t^\top (f_t - y)$$

where

$$\tilde{\Phi}_t = \left(-\nabla_\theta f(x_i; \theta_t)^\top -\right)_{i\le n} \in \mathbb{R}^{n\times|\theta|} \quad \text{and} \quad f_t = (f(x_i; \theta_t))_{i\le n} \in \mathbb{R}^n$$

Here $\tilde{\Phi}_t$ is the feature matrix whose i-th row is the gradient of the network output with respect to all parameters, evaluated at data point x_i and current parameters θ_t.

In the *lazy training regime* (small learning rate, wide network), the parameters stay close to initialization, so we can linearize the network via a first-order Taylor expansion around θ_0:

$$f_t \approx f_0 + \tilde{\Phi}_0 (\theta_t - \theta_0)$$

Assuming the feature matrix $\tilde{\Phi}_t$ remains approximately constant at $\tilde{\Phi}_0$ (which holds when $m \to \infty$), gradient flow yields

$$\theta_t - \theta_0 = \tilde{\Phi}_0^\top \left(\tilde{\Phi}_0 \tilde{\Phi}_0^\top\right)^{-1} (f_t - f_0)$$

This says the parameter change lies in the span of the gradients and is chosen to optimally fit the training residuals.

Now let x_* be a test point. Plugging the linearization into the prediction yields

$$f_t(x_*) \approx f_0(x_*) + \nabla_\theta f(x_*, \theta_0)^\top \tilde{\Phi}_0^\top \left(\tilde{\Phi}_0 \tilde{\Phi}_0^\top\right)^{-1} (f_t - f_0)$$

Define the *neural tangent kernel (NTK)* K with entries

$$K_{ij} = \nabla_\theta f(x_i, \theta_0)^\top \nabla_\theta f(x_j, \theta_0) = \left[\tilde{\Phi}_0 \tilde{\Phi}_0^\top\right]_{ij}$$

and similarly $k(x_*) = \left(\nabla_\theta f(x_*, \theta_0)^\top \nabla_\theta f(x_i, \theta_0)\right)_{i \le n}$.

In the infinite-width limit ($m \to \infty$), the random initialization ensures $f_0(x) \to 0$ for all x (the outputs average out), and after infinite training time ($t \to \infty$), gradient descent drives the training residual to zero so $f_t \to y$. The prediction becomes

$$f_\infty(x_*) = k(x_*)^\top K^{-1} y$$

This is exactly the result we obtained in Act 3 for kernel regression.

Conclusion: Gradient descent on a very wide neural network operates in a *kernel regime*, where training is equivalent to kernel ridge regression with the neural tangent kernel. The NTK is determined by the architecture and activation function, but the solution has the same closed-form structure $k(x_*)^\top K^{-1} y$ as any other kernel method. In practice, finite-width networks can escape this regime and learn richer, feature-learning representations—this lazy limit is a useful theoretical baseline.

4.5 Some Things from the Slides

MAP, ERM, and the conditional mean. For squared loss and any hypothesis class rich enough to contain the regression function, the Bayes-optimal solution is the conditional expectation $f^\star(x) = \mathbb{E}[Y \mid X = x]$, i.e.

$$f^\star \in \arg\min_f \mathbb{E}_{X,Y}\left[(Y - f(X))^2\right].$$

In practice $P(Y \mid X)$ is unknown, so we either (i) postulate a parametric model and perform maximum likelihood / MAP (e.g., assume $Y \mid X \sim \mathcal{N}(f_\beta(X), \sigma^2)$ and maximize $\sum_i \log p(y_i \mid x_i, \beta)$) or (ii) minimize the empirical risk $\sum_i (y_i - f(x_i))^2$ directly. For well-behaved models these two routes coincide, which explains why classical ERM with squared loss reproduces the MAP estimator of a Gaussian noise model.

Gauss–Markov optimality. Ordinary least squares does not merely give *a* solution, it gives the best linear unbiased estimator (BLUE). Consider any linear estimator $\tilde{\theta} = c^\top y$ that is unbiased for $a^\top \beta$ (i.e., $\mathbb{E}[\tilde{\theta}] = a^\top \beta$). The Gauss–Markov theorem states

$$\mathrm{Var}(a^\top \hat{\beta}) \le \mathrm{Var}(\tilde{\theta}),$$

where $\hat{\beta} = (X^\top X)^{-1} X^\top y$ is the OLS solution and a is an arbitrary vector. Intuitively, any other unbiased linear estimator differs from OLS by an additive linear operator D with $a^\top D X = 0$, which only inflates variance through the positive semi-definite term $DD^\top$. This reinforces the motivation for OLS (or its regularized cousins) when unbiasedness and linearity are desired.

Bias–variance decomposition. Suppose we highlight a specific input x and view the learned regressor $\hat{f}$ as a random variable (due to sampling various training sets). The expected squared prediction error decomposes into variance, bias, and irreducible noise:

$$\mathbb{E}_D \mathbb{E}_{Y|X=x}\left[(\hat{f}(x) - Y)^2\right] = \underbrace{\mathbb{E}_D\left[(\hat{f}(x) - \mathbb{E}_D[\hat{f}(x)])^2\right]}_{\text{variance}} + \underbrace{\left(\mathbb{E}_D[\hat{f}(x)] - \mathbb{E}[Y \mid X = x]\right)^2}_{\text{bias}^2} + \underbrace{\mathrm{Var}(Y \mid X = x)}_{\text{noise}}.$$

Low-capacity models (small hypothesis classes) produce high bias and low variance, while expressive models produce the opposite. Managing this trade-off is the crux of regularization and model selection.

Shrinkage beyond ridge and lasso. Ridge (ℓ_2) and lasso (ℓ_1) penalties are instances of a broader shrinkage family

$$\hat{\beta} = \arg\min_\beta \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^d x_{i,j}\beta_j\right)^2 + \lambda \sum_{j=1}^d |\beta_j|^q, \quad q \in (0, \infty].$$

Varying q changes the geometry of the constraint set: $q = 2$ yields spherical ridge contours, $q = 1$ produces diamond-shaped lasso constraints whose corners encourage sparsity, and $q < 1$

(non-convex) promotes even stronger sparsity at the cost of more difficult optimization. These shrinkage priors can be interpreted as MAP estimators with different prior distributions on β (Gaussian, Laplace, etc.) and help calibrate the bias–variance compromise by shrinking noisy coefficients toward zero. In practice we trace the coefficient paths as the tuning parameter (either λ or the effective degrees of freedom) varies, and select the desired amount of shrinkage via cross-validation on held-out data. Ridge paths shrink smoothly without hitting zero, whereas lasso paths *do* cross zero, enabling feature selection alongside regularization.

5 Representations

Machine learning algorithms only see the world through the representations we choose. Good representations clarify the objective (expected risk), guide the empirical procedures we rely on (empirical risk and test evaluation), and encode the structure of the data via adequate feature, scale, and mathematical spaces.

5.1 Expected vs. empirical risk

Given a hypothesis $f \in \mathcal{C}$, a loss function Q and random variables (X, Y), the conditional expected risk is

$$R(f, X) = \int Q(Y, f(X))\, \mathbb{P}(Y \mid X)\, dY,$$

while the total expected risk integrates over the data distribution

$$R(f) = \int R(f, X)\, \mathbb{P}(X)\, dX = \iint Q(Y, f(X))\, \mathbb{P}(X, Y)\, dX\, dY.$$

Learning is phrased as minimizing $R(f)$, but we only have data. With a training sample $Z_{\text{train}} = \{(X_i, Y_i)\}_{i=1}^{n}$, the empirical risk (training error) of an estimator $\hat{f}_n$ is

$$\widehat{R}(\hat{f}_n, Z_{\text{train}}) = \frac{1}{n} \sum_{i=1}^{n} Q(Y_i, \hat{f}_n(X_i)), \qquad \hat{f}_n \in \arg\min_{f \in \mathcal{C}} \widehat{R}(f, Z_{\text{train}}).$$

The test set $Z_{\text{test}} = \{(X_j, Y_j)\}_{j=n+1}^{n+m}$ yields an unbiased estimate of $R(\hat{f}_n)$ *only* if it is held out until the final estimator is fixed:

$$\widehat{R}(\hat{f}_n, Z_{\text{test}}) = \frac{1}{m} \sum_{j=n+1}^{n+m} Q(Y_j, \hat{f}_n(X_j)).$$

Whenever we adapt the estimator or its hyperparameters using the test set, we introduce dependencies between the model and the data and obtain a too optimistic risk estimate. Statistical learning therefore mandates the strict separation of training, validation, and testing.

5.2 Comparing algorithms on shared test data

To compare two learning algorithms $A^{(I)}$ and $A^{(II)}$ on the same dataset, we evaluate their paired losses on each test point j:

$$\Delta_j = Q(Y_j, \hat{f}_n^{(I)}(X_j)) - Q(Y_j, \hat{f}_n^{(II)}(X_j)), \qquad j = n+1, \ldots, n+m.$$

The sample mean $\overline{\Delta}$ and standard deviation $\text{std}(\Delta)$ of these paired differences quantify which model is better. If $\overline{\Delta} - 2\,\text{std}(\Delta) > 0$, then algorithm $A^{(II)}$ is reliably superior to $A^{(I)}$. This "first compare, then average" rule mirrors paired t-tests and guards against spurious conclusions from aggregate metrics alone.

5.3 Data, feature, and measurement spaces

Representation begins with deciding *what* we measure:

- **Object space $\mathcal{O}$.** Objects (digits, patients, sounds) form the domain of discourse.
- **Measurements X.** A measurement maps tuples of objects into a codomain K: $X : \mathcal{O}^{(1)} \times \cdots \times \mathcal{O}^{(R)} \to K$. Measurements can be direct (pixel intensities) or derived (edges, mel-cepstral coefficients).
- **Feature space $\mathcal{X}$.** Selecting $\mathcal{X}$ fixes the admissible metric and invariances. Numeric $\mathbb{R}^d$, Boolean, or categorical spaces each encode different similarity notions.

Typical data types emerge from the arity of the measurement:

Monadic/vectorial: $X : \mathcal{O} \to \mathbb{R}^d$ for temperature maps or feature vectors.

Dyadic: $X : \mathcal{O}^{(1)} \times \mathcal{O}^{(2)} \to \mathbb{R}$ for user–item interactions or contingency tables.

Pairwise similarity: $X : \mathcal{O} \times \mathcal{O} \to \mathbb{R}$ for protein alignment scores or image patch similarities.

Polyadic/multiway: $X : \mathcal{O}^{(1)} \times \mathcal{O}^{(2)} \times \mathcal{O}^{(3)} \to \mathbb{R}$ such as person × behavior × trait tensors or preferential choice data.

The taxonomy clarifies whether the learner should expect absolute values, co-occurrence counts, or structured relational inputs.

5.4 Scale types and transformation invariances

Scale choices express which transformations should leave conclusions invariant:

- **Nominal scale:** only equality matters; any bijection preserves meaning.
- **Ordinal scale:** rankings are meaningful; order-preserving functions are admissible.
- **Interval scale:** information resides in differences; affine transformations $f(x) = ax + c$ with $a > 0$ are allowed (e.g., Fahrenheit).
- **Ratio scale:** differences and ratios matter; only scalings $f(x) = ax$, $a > 0$ preserve structure (e.g., Kelvin).
- **Absolute scale:** literal values matter; only the identity transformation is valid (e.g., exam grades).

Data whitening—scaling features by their standard deviation—is one practical way to enforce comparable dynamic ranges so that the chosen metric respects the intended invariances.

5.5 Mathematical spaces underlying representations

Different mathematical spaces formalize increasingly rich structures:

- **Topological spaces** $(X, \mathcal{J})$**:** $\mathcal{J}$ is a family of subsets containing X and $\varnothing$, closed under finite intersections and arbitrary unions. Topology captures neighborhood relations without prescribing distances.
- **Metric spaces** (X, d)**:** A metric d satisfies non-negativity, identity of indiscernibles, symmetry, and the triangle inequality (or the stronger ultrametric inequality). Metrics quantify distances and therefore induce topologies.
- **Euclidean vector spaces** (V, ϕ)**:** A vector space equipped with a scalar product ϕ satisfying distributivity, symmetry, homogeneity, and positive definiteness. The induced norm $\|x\| = \sqrt{\phi(x, x)}$ generalizes standard Euclidean geometry.

Every Euclidean space is metric and thus topological, but the converse need not hold. Representation design should match the amount of reliable structure (topological, metric, or vectorial) that the measurements truly support.

5.6 Probability spaces

Probability theory provides the formal language for risk:

- **Sample space** $\Omega = \{\omega_1, \ldots, \omega_N\}$ collects all elementary outcomes (e.g., sequences of coin flips).
- **Event algebra** $\mathcal{A} \subseteq 2^\Omega$ contains Ω and is closed under union, intersection, and set difference, so that compound statements about outcomes remain valid events.

- **Probability measure** $\mathbb{P} : \mathcal{A} \to [0, 1]$ assigns weights $p(\omega_i)$ to elementary events, satisfies $\mathbb{P}(\Omega) = 1$, and extends additively to events $A \in \mathcal{A}$ via $\mathbb{P}(A) = \sum_{\omega_i \in A} p(\omega_i)$.

The triple $(\Omega, \mathcal{A}, P)$ underpins expected risk: once we define the events (representations) and their probabilities, we can integrate losses and reason about generalization.

6 Gaussian Processes for Regression

Gaussian processes (GPs) turn Bayesian linear regression into a flexible, non-parametric model that reasons about functions via distributions over all possible outputs. The slides emphasize three pillars: (i) the Bayesian linear regression foundation, (ii) kernel engineering for encoding inductive biases, and (iii) prediction, validation, and application pipelines that exploit GP uncertainty.

6.1 From Bayesian linear regression to GPs

Linear regression with Gaussian noise assumes $Y = x^\top \beta + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Placing a Gaussian prior on the weights, $\beta \sim \mathcal{N}(0, \Lambda^{-1})$, yields the posterior

$$p(\beta \mid X, y) = \mathcal{N}\big(\mu_\beta, \Sigma_\beta\big), \qquad \mu_\beta = \big(X^\top X + \sigma^2 \Lambda\big)^{-1} X^\top y, \quad \Sigma_\beta = \sigma^2 \big(X^\top X + \sigma^2 \Lambda\big)^{-1}.$$

Understanding the posterior: The posterior mean μ_β combines the data (via $X^\top X$ and $X^\top y$) with the prior precision Λ. This is exactly the ridge regression solution we saw in Chapter 4.2, where Λ acts as a regularization matrix (compare to equation 1). The term $X^\top X + \sigma^2 \Lambda$ plays the role of a regularized Hessian, stabilizing the inversion. The posterior covariance Σ_β (compare to equation 2) captures uncertainty about β: directions along which the data is uninformative (small eigenvalues of $X^\top X$) retain their prior uncertainty, while directions well-explained by data shrink toward zero. Note that all uncertainty decreases with sample size n (more rows in X), and the marginal posterior variance of the i-th weight is Σ_β^{ii}.

Observations are linear combinations of Gaussian variables, so the vector of outputs $y = X\beta + \varepsilon$ is jointly Gaussian with mean zero and covariance

$$\mathrm{Cov}(y) = X \Lambda^{-1} X^\top + \sigma^2 I_n.$$

Derivation. Since $\beta \sim \mathcal{N}(0, \Lambda^{-1})$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$ are independent,

$$\begin{aligned}
\mathbb{E}[y] &= \mathbb{E}[X\beta + \varepsilon] = X\,\mathbb{E}[\beta] + \mathbb{E}[\varepsilon] = 0, \\
\mathrm{Cov}(y) &= \mathbb{E}\big[(y - \mathbb{E}y)(y - \mathbb{E}y)^\top\big] = \mathbb{E}\big[(X\beta + \varepsilon)(X\beta + \varepsilon)^\top\big] \\
&= X\,\mathbb{E}[\beta\beta^\top] X^\top + \mathbb{E}[\varepsilon\varepsilon^\top] + X\,\mathbb{E}[\beta\varepsilon^\top] + \mathbb{E}[\varepsilon\beta^\top] X^\top \\
&= X\,\Lambda^{-1} X^\top + \sigma^2 I_n,
\end{aligned}$$

where the cross terms vanish by independence (and zero means).

Defining $k(x_i, x_j) = x_i^\top \Lambda^{-1} x_j$ turns this covariance into a kernel (Gram) matrix K, so $y \sim \mathcal{N}(0, K + \sigma^2 I)$. Replacing the dot-product kernel by any valid positive semi-definite kernel function "kernelizes" Bayesian ridge regression; the resulting stochastic process over functions is a Gaussian process.

6.2 What is a Gaussian process?

A **Gaussian process (GP)** is a collection of random variables indexed by inputs (e.g., $x \in \mathbb{R}^d$) such that any finite subset has a joint Gaussian distribution. Equivalently, a GP is a *distribution over functions*. We write

$$f \sim \mathcal{GP}\big(m, k\big), \quad m(x) = \mathbb{E}[f(x)], \quad k(x, x') = \mathrm{Cov}(f(x), f(x')).$$

For any inputs $X = [x_1, \ldots, x_n]$, the function values satisfy

$$f(X) = [f(x_1), \ldots, f(x_n)]^\top \sim \mathcal{N}\big(m(X),\ K\big), \quad K_{ij} = k(x_i, x_j).$$

With Gaussian observation noise $y = f(X) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$, we obtain

$$y \sim \mathcal{N}\big(m(X),\ K + \sigma^2 I\big).$$

The mean function m and kernel k fully specify the prior over functions: the kernel encodes smoothness, invariances, and correlation structure. Conditioning the prior GP on data (X, y) yields a *posterior GP* with updated mean and covariance, whose predictive mean/variance coincide with the closed-form formulas presented below. This makes GPs a principled, non-parametric way to model functions with calibrated uncertainty.

6.3 Kernel design

Kernels encode similarity and thereby the structure of the functions we wish to learn:

- Valid kernels must be symmetric and generate positive semi-definite Gram matrices for any finite set of inputs. They implicitly act as inner products in (possibly infinite-dimensional) Hilbert spaces.
- Standard kernels on $\mathbb{R}^d$ include linear $k(x, x') = x^\top x'$, polynomial $k(x, x') = (x^\top x' + 1)^p$, squared exponential $k(x, x') = \sigma_f^2 \exp(-\|x - x'\|^2/(2\ell^2))$, rational quadratic, exponential, and periodic kernels. Each carries different invariances (smoothness, periodicity, etc.).
- Kernels compose: sums, products, positive scalings, or applying positive-coefficient polynomials / exponentials to a base kernel all preserve validity. This "kernel engineering" enables similarity measures on non-vector data such as strings, graphs (diffusion kernels), or probability distributions.

Designing an appropriate kernel is tantamount to choosing the hypothesis space for the GP.

6.4 Prediction with Gaussian processes

Given training data (X, y) and a test input $x_\star$, we are interested in $y_\star$. The joint distribution of y and $y_\star$ is Gaussian:

$$\begin{bmatrix} y \\ y_\star \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K + \sigma^2 I & k \\ k^\top & c \end{bmatrix}\right),$$

where $K = k(X, X)$, $k = k(x_\star, X)$ is the vector of cross-covariances, and $c = k(x_\star, x_\star) + \sigma^2$. This is a natural extension of the setup without $x_\star$. Conditioning a multivariate Gaussian gives the predictive distribution

$$p(y_\star \mid x_\star, X, y) = \mathcal{N}\big(\mu_\star, \sigma_\star^2\big), \quad \mu_\star = k^\top (K + \sigma^2 I)^{-1} y, \quad \sigma_\star^2 = c - k^\top (K + \sigma^2 I)^{-1} k.$$

Derivation. **Theorem (Conditional Gaussian).** Suppose

$$\begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right),$$

with $\mathbf{a}_1, \mathbf{u}_1 \in \mathbb{R}^e$, $\mathbf{a}_2, \mathbf{u}_2 \in \mathbb{R}^f$, and covariance blocks $\Sigma_{11} \in \mathbb{R}^{e\times e}$, $\Sigma_{12} \in \mathbb{R}^{e\times f}$, $\Sigma_{21} \in \mathbb{R}^{f\times e}$, $\Sigma_{22} \in \mathbb{R}^{f\times f}$ positive semidefinite. Then the conditional distribution of $\mathbf{a}_2$ given $\mathbf{a}_1 = \mathbf{z}$ is

$$p(\mathbf{a}_2 \mid \mathbf{a}_1 = \mathbf{z}) = \mathcal{N}\big(\mathbf{u}_2 + \Sigma_{21}\Sigma_{11}^{-1}(\mathbf{z} - \mathbf{u}_1),\ \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\big).$$

Hence GPs output **both a mean prediction and an uncertainty quantification**. This is however not the real uncertainty about the function value at $x_\star$, but the uncertainty we get inside our model, which is constrained to gaussian processes with a specific kernel.

Algorithm Prediction with Gaussian Processes

Require: n observed data ($\mathbf{X} = (x_1, \ldots, x_n)^\mathsf{T} \in \mathbb{R}^{n\times d}$, $\mathbf{y} \in \mathbb{R}^n$), kernel function k, noise variance σ^2, new data point $x_{n+1} \in \mathbb{R}^d$

$\mathbf{K} \leftarrow (k(x_i, x_j))_{1\le i,j\le n}$ // Compute kernel matrix

$\mathbf{k} \leftarrow (k(x_{n+1}, x_i))_{1\le i\le n}$ // Similarity of new data point and observed data

$\mu_{y_{n+1}} \leftarrow \mathbf{k}^\mathsf{T}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$ // Mean of predictive distribution

$\sigma^2_{y_{n+1}} \leftarrow k(x_{n+1}, x_{n+1}) - \mathbf{k}^\mathsf{T}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}$ // Variance of predictive distribution

return $\mathcal{N}(y_{n+1} | \mu_{y_{n+1}}, \sigma^2_{y_{n+1}})$ // Return predictive distribution

The prediction algorithm returns a distribution function. The prediction at x_{n+1} yields a mean value $\mu_{y_{n+1}}$ and a variance $\sigma^2_{y_{n+1}}$. Furthermore, samples y_{n+1} can be drawn from this distribution.

6.5 Validating kernels and hyperparameters

Practical GP performance depends on hyperparameters such as the length-scale ℓ, signal variance σ_f^2, or kernel choice. The lecture highlights:

- Evidence maximization (type-II ML) to optimize hyperparameters by maximizing $\log p(y \mid X, \theta)$.
- Cross-validation schemes (random splits, leave-one-out) that rank kernels by predictive performance on held-out data. Synthetic experiments show that proper scoring rules recover the data-generating kernel, while real data (e.g., power plant energy output) can exhibit different optima depending on the scoring metric (squared exponential vs. periodic kernels).
- Bayesian comparison of "teacher" and "student" kernels: evaluate how well a candidate kernel matches data generated from another kernel by comparing their posterior predictive distributions.

Kernel validation is therefore an empirical model-selection layer that complements theoretical kernel properties.

6.6 Applications: control and fMRI

Because GPs model functions with calibrated uncertainty, they are appealing for safety-critical and data-efficient applications:

- **Safe Bayesian optimization for control.** In automatic controller tuning (e.g., quadrotor flight), the unknown performance function over controller parameters is modeled as a GP. Safe optimization explores only parameter settings whose predicted performance exceeds a safety threshold with high probability, gradually expanding the safe set and converging to the global optimum using few evaluations.
- **Robust controller design.** GPs capture both the mean and variance of system responses, enabling controllers that respect safety constraints while improving performance across uncertain dynamics. Comparisons to hand-tuned controllers highlight faster convergence and higher reliability.

These case studies reinforce that GP regression is more than a curve fitting tool—it is a probabilistic modeling framework that unifies inference, kernel engineering, and safety-aware decision making.

7 Ensemble Methods

Ensemble methods combine multiple base learners to obtain a predictor whose bias, variance, or loss surface is superior to any constituent model. By averaging or sequentially correcting learners trained on diverse views of the data, ensembles produce more stable predictions, calibrated uncertainties, and richer inductive biases than single models. This chapter summarizes the bootstrap-based family (bagging and random forests), boosting, and more general stacking strategies, with an emphasis on how they target different points along the bias–variance trade-off.

7.1 Why ensembles help

Let $\hat{f}(x)$ be a single hypothesis trained on data D. Ensembles form $\hat{f}_{\text{ens}}(x) = \sum_{b=1}^{B} w_b \hat{f}_b(x)$ from base learners $\hat{f}_b$. Two canonical effects explain their success:

- **Variance reduction.** If the $\hat{f}_b$'s are identically distributed with variance σ^2 and pairwise correlation ρ, then
$$\operatorname{Var}\big[\hat{f}_{\text{avg}}(x)\big] = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2,$$
which shrinks as B increases whenever $\rho < 1$. Bagging manipulates the data (bootstrap samples) to drive correlations down and thus stabilize high-variance learners such as decision trees or neural networks trained on small data.

  Derivation. For the average predictor $\hat{f}_{\text{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x)$,
$$\begin{aligned}
\operatorname{Var}[\hat{f}(x)] &= \mathbb{E}_D\left(\hat{f}(X) - \mathbb{E}_D\hat{f}(X)\right)^2 \\
&= \mathbb{E}_D\left(\frac{1}{B}\sum_{i=1}^{B}\hat{f}_i(x) - \frac{1}{B}\sum_{i=1}^{B}\mathbb{E}_D\hat{f}_i(x)\right)^2 \\
&= \mathbb{E}_D\left(\frac{1}{B}\sum_{i=1}^{B}\left(\hat{f}_i(x) - \mathbb{E}_D\hat{f}_i(x)\right)\right)^2 \\
&= \frac{1}{B^2}\sum_{i=1}^{B}\operatorname{Var}_D\left[\hat{f}_i(x)\right] + \frac{1}{B^2}\sum_{i\neq j}\operatorname{Cov}\left[\hat{f}_i(x), \hat{f}_j(x)\right]
\end{aligned}$$
  If all base learners share variance σ^2 and pairwise covariance $\operatorname{Cov}(\hat{f}_i, \hat{f}_j) = \rho\,\sigma^2$,
$$\begin{aligned}
\operatorname{Var}\big[\hat{f}_{\text{avg}}(x)\big] &= \frac{B}{B^2}\sigma^2 + \frac{B(B-1)}{B^2}\,\rho\,\sigma^2 \\
&= \frac{1}{B}\,\sigma^2 + \left(1 - \frac{1}{B}\right)\rho\,\sigma^2 \\
&= \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.
\end{aligned}$$
  Thus variance decreases like $1/B$ when correlations are small ($\rho \approx 0$), while residual correlation limits the gain.

- **Bias correction.** Sequential ensembles such as boosting fit a series of weak learners to the residuals (negative gradients) of the current model. Each stage nudges the predictor toward lower bias and can transform a barely better-than-random base learner into a strong classifier.

These mechanisms are complementary: bagging primarily attacks variance, boosting primarily attacks bias, and stacking blends heterogeneous models to capture complementary inductive biases.

7.2 Bagging

Bootstrap aggregating (bagging) trains B base predictors on bootstrap replicates D_b drawn with replacement from D. For a regression tree learner $\hat{f}(x; D_b)$ the bagged predictor is

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}(x; D_b), \qquad \hat{p}_{\text{bag}}(y \mid x) = \frac{1}{B}\sum_{b=1}^{B} \hat{p}(y \mid x; D_b)$$

for regression or probabilistic classification, respectively. Averaging smooths out spurious splits triggered by sampling noise and produces nearly unbiased uncertainty estimates by comparing the ensemble spread around $\hat{f}_{\text{bag}}$.

Out-of-bag (OOB) validation. Each data point is excluded from roughly 36% of bootstrap samples. Predictions aggregated over models that did not train on x_i approximate leave-one-out validation without an explicit validation set, providing error estimates and enabling tuning (e.g., tree depth, number of features) directly from training data.

Note. Bagging works best when the base learner is (i) high variance and (ii) able to fit the bootstrap sample strongly. Stable learners (e.g., linear regression with ℓ_2 regularization) gain little because resampling barely changes the fitted model, so the ensemble collapses to a single hypothesis.

7.3 Random forests

Random forests enhance bagging by injecting feature-level randomness, thereby decorrelating the trees further. In each internal node, a tree chooses the best split among m_{try} randomly selected features (commonly $\sqrt{d}$ for classification or $d/3$ for regression). The resulting predictor

$$\hat{f}_{\text{RF}}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x),$$

where T_b is a fully grown, unpruned tree, inherits low variance because the node-level randomness reduces ρ in the variance expression above. Additional benefits include:

- **Implicit feature selection:** features rarely chosen for splits likely lack predictive power; aggregating split statistics yields variable-importance measures.
- **Robustness to class imbalance:** by re-weighting bootstrap draws or using balanced subsampling, forests maintain accuracy even when one class dominates.
- **Proximities and uncertainty:** the fraction of trees that land in the same leaf as x_i defines a similarity metric; empirical distributions of votes supply predictive intervals.

7.4 Boosting

Boosting constructs an additive model

$$F_T(x) = \sum_{t=1}^{T} \gamma_t h_t(x)$$

with weak learners h_t chosen to maximize the descent along the negative gradient of a differentiable loss. AdaBoost specializes this framework to exponential loss and binary classification:

1. Initialize sample weights $w_i^{(1)} = 1/n$.
2. At step t, train h_t to minimize weighted error $\varepsilon_t = \sum_i w_i^{(t)} \mathbf{1}\{h_t(x_i) \neq y_i\}$.
3. Set $\alpha_t = \frac{1}{2}\log\frac{1-\varepsilon_t}{\varepsilon_t}$ and update weights

$$w_i^{(t+1)} = \frac{w_i^{(t)} \exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$

where Z_t normalizes the distribution.

4. Output $F_T(x) = \operatorname{sign}\Big(\sum_{t=1}^{T} \alpha_t h_t(x)\Big)$.

Each stage focuses on examples misclassified so far. Under mild conditions, training error decays exponentially with T, and the margin distribution explains AdaBoost's generalization ability.

Gradient boosting generalizes AdaBoost by performing steepest-descent in function space. For differentiable loss $\ell(y, F(x))$, the residuals are the negative gradients $r_i^{(t)} = -\left.\frac{\partial \ell(y_i, F(x_i))}{\partial F}\right|_{F=F_{t-1}}$. Fitting a regression tree h_t to $(x_i, r_i^{(t)})$ and adding it with learning rate ν yields

$$F_t(x) = F_{t-1}(x) + \nu \gamma_t h_t(x), \qquad \gamma_t = \arg\min_{\gamma} \sum_i \ell\left(y_i, F_{t-1}(x_i) + \nu \gamma h_t(x_i)\right).$$

Choice of ν, tree depth, and subsampling ratio controls shrinkage and prevents overfitting. Modern variants (XGBoost, LightGBM, CatBoost) incorporate second-order Taylor approximations, histogram-based splits, and column subsampling to scale boosting to millions of observations.

7.5 Stacking and model averaging

Stacked generalization (stacking) learns a meta-model on top of out-of-fold predictions from heterogeneous base learners such as neural networks, kernel machines, and tree ensembles. Given base predictions $z_i = [\hat{f}_1(x_i), \ldots, \hat{f}_K(x_i)]^\top$, a meta-learner $g(z)$ minimizes validation loss and outputs

$$\hat{y} = g\big(\hat{f}_1(x), \ldots, \hat{f}_K(x)\big).$$

Cross-validation prevents target leakage by ensuring g never sees predictions on points used to fit the corresponding base model. Compared to bagging or boosting, stacking can exploit complementary feature representations (e.g., CNN features plus hand-crafted statistics) and yields calibrated probabilistic forecasts when g is a logistic regression with constraints $\sum_k w_k = 1$, $w_k \geq 0$.

7.6 Practical guidance

- **Hyperparameter selection.** Number of estimators (B), tree depth, learning rate, and subsampling ratios have the largest impact. OOB error supplies an inexpensive proxy for cross-validation in bagging/forests, whereas boosting benefits from k-fold or time-series splits.
- **Interpretability.** Partial dependence plots and SHAP values can be computed on ensembles to interpret non-linear interactions. Random forest feature importance or permutation scores act as a first diagnostic, but they may be biased toward high-cardinality features; conditional permutation mitigates this.
- **Computation.** Parallelize bagging and random forests over trees; boosting is sequential but can parallelize histogram construction and tree search. Warm-starting (re-using previous trees) accelerates hyperparameter sweeps.
- **Regularization.** Early stopping, shrinkage ($\nu < 0.1$), ℓ_1/ℓ_2 penalties on leaf weights, or sample/feature subsampling keep boosting ensembles from overfitting. For bagging, limiting tree depth or minimum leaf size guards against noise amplification.

Ensembles thus provide a versatile toolbox: use bagging and random forests when variance is the bottleneck, boosting when bias dominates or when precise control over loss gradients is needed, and stacking when heterogeneous models capture distinct structures in the data.

8 Convex Optimization

Convex optimization studies problems whose objective and feasible sets are convex, guaranteeing that every local optimum is also global. These structure-driven guarantees let us design algorithms with provable convergence, quantify sensitivity, and provide certificates of optimality via dual variables. This chapter outlines the key definitions, canonical problem classes, duality theory, and foundational algorithms emphasized in the lecture notes.

8.1 Convex sets, functions, and problems

Let $C \subseteq \mathbb{R}^d$ be a set. C is **convex** if for any $x, y \in C$ and $\theta \in [0,1]$, the convex combination $\theta x + (1-\theta)y$ lies in C. Typical convex sets are affine subspaces, halfspaces, Euclidean balls, ellipsoids, spectrahedra, and probability simplices.

A function $f : \mathbb{R}^d \to \mathbb{R}$ is **convex** if its domain is convex and for every x, y and $\theta \in [0,1]$

$$f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta) f(y).$$

When f is differentiable, convexity is equivalent to the first-order inequality

$$f(y) \ge f(x) + \nabla f(x)^\top (y - x),$$

meaning the affine tangent is a global under-estimator. If f is twice differentiable, convexity is equivalent to $\nabla^2 f(x) \succeq 0$ for all x in the interior of the domain. **Strong convexity** with parameter $m > 0$ strengthens the inequality to

$$f(y) \ge f(x) + \nabla f(x)^\top (y - x) + \frac{m}{2}\|y - x\|^2,$$

implying a unique minimizer and improved convergence rates.

A generic convex optimization problem reads

$$\begin{aligned} \min_{x \in \mathbb{R}^d} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \le 0, \quad i = 1, \dots, m, \\ & Ax = b, \end{aligned}$$

where each f_i is convex and the equality constraints define an affine set. The feasible set $\{x \mid f_i(x) \le 0, Ax = b\}$ must be nonempty for the problem to be well-posed. Convexity ensures that any Karush–Kuhn–Tucker (KKT) point is globally optimal.

8.2 Standard convex programs in machine learning

Many estimators from earlier chapters fit this template:

- **Least squares / ridge regression:** $f(x) = \frac{1}{2}\|Ax - b\|_2^2 + \frac{\lambda}{2}\|x\|_2^2$ is convex quadratic; ridge adds strong convexity, yielding the closed form in equation (1).
- **Lasso:** minimize $\frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1$. The ℓ_1 norm promotes sparsity via a polyhedral penalty while keeping the problem convex. Soft-thresholding is the proximal operator driving coordinate descent.
- **Support Vector Machines (SVM):** hinge-loss objectives $\sum_i \max(0, 1 - y_i w^\top x_i)$ with ℓ_2 regularization define convex problems whose dual has sparse support vectors.
- **Logistic regression:** the negative log-likelihood $\sum_i \log(1+\exp(-y_i w^\top x_i))$ is convex. Adding ℓ_1 or ℓ_2 penalties yields generalized linear models solvable via gradient or Newton methods.
- **Matrix completion:** minimizing $\sum_{(i,j)\in\Omega}(X_{ij} - M_{ij})^2 + \lambda\|X\|_*$ uses the nuclear norm (convex surrogate of rank) to recover low-rank matrices.

Specialized subfamilies often admit faster algorithms:

- **Linear programs (LP):** $f_0(x) = c^\top x$, $f_i(x) = a_i^\top x - b_i$.
- **Quadratic programs (QP):** quadratic objective with linear constraints.
- **Second-order cone programs (SOCP)** and **semidefinite programs (SDP)** capture norms and PSD constraints, respectively, and power robust control plus covariance fitting.

8.3 Lagrangian duality and KKT conditions

For

$$\min_x f_0(x) \quad \text{s.t.} \quad f_i(x) \le 0,\ Ax = b,$$

introduce multiplier $\lambda \ge 0$ for inequalities and ν for equalities. The **Lagrangian**

$$\mathcal{L}(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \nu^\top (Ax - b)$$

lower-bounds the primal objective: $g(\lambda, \nu) = \inf_x \mathcal{L}(x, \lambda, \nu) \le f_0(x)$ for all feasible x. The **dual problem** maximizes $g(\lambda, \nu)$ subject to $\lambda \ge 0$. Weak duality ($g \le p^\star$) always holds, while **strong duality** ($g^\star = p^\star$) holds under mild constraint qualifications such as Slater's condition (strict feasibility). Dual variables often have sensitivity interpretations, e.g., the effect of tightening constraints.

The **KKT conditions** describe optimality when strong duality holds:

$$\begin{aligned}
&\text{Primal feasibility: } f_i(x^\star) \le 0,\ Ax^\star = b.\\
&\text{Dual feasibility: } \lambda^\star \ge 0.\\
&\text{Stationarity: } \nabla f_0(x^\star) + \sum_i \lambda_i^\star \nabla f_i(x^\star) + A^\top \nu^\star = 0.\\
&\text{Complementary slackness: } \lambda_i^\star f_i(x^\star) = 0.
\end{aligned}$$

For unconstrained problems these reduce to $\nabla f_0(x^\star) = 0$, consistent with calculus. In SVMs, for example, KKT implies that only points on the margin have non-zero dual variables, explaining sparsity in the dual solution.

8.4 First-order algorithms

Most large-scale machine learning problems rely on first-order (gradient-based) methods with low per-iteration cost.

- **Gradient descent (GD).** Iterate $x_{k+1} = x_k - \eta_k \nabla f(x_k)$. For L-smooth convex f, constant step-size $\eta = 1/L$ yields $f(x_k) - f^\star \le \frac{L\|x_0 - x^\star\|^2}{2k}$. If f is additionally m-strongly convex, the error contracts linearly: $f(x_k) - f^\star \le \left(1 - \frac{m}{L}\right)^k (f(x_0) - f^\star)$.
- **Projected gradient.** For constrained problems with easy projections Π_C, update $x_{k+1} = \Pi_C(x_k - \eta \nabla f(x_k))$. This applies to simplex constraints, ℓ_2-balls, or PSD cones (via eigenvalue thresholding).
- **Subgradient methods.** When f is convex but nonsmooth (e.g., hinge or ℓ_1 norms), use $x_{k+1} = x_k - \eta g_k$ with $g_k \in \partial f(x_k)$. Convergence is $O(1/\sqrt{k})$ for general convex functions.
- **Accelerated methods.** Nesterov's accelerated gradient adds a momentum term achieving $f(x_k) - f^\star = O(1/k^2)$ for L-smooth convex objectives, approaching optimal complexity for first-order methods.
- **Stochastic gradients.** SGD and its variants (momentum, Adam, Adagrad) replace ∇f with unbiased estimates, giving scalable training on massive datasets. Convexity enables convergence proofs, e.g., $\mathbb{E}[f(\bar{x}_k)] - f^\star = O(1/\sqrt{k})$ with diminishing steps.

8.5 Second-order and interior-point methods

Second-order methods leverage curvature for faster local convergence.

- **Newton's method.** Update $x_{k+1} = x_k - \nabla^2 f(x_k)^{-1} \nabla f(x_k)$. Under mild conditions, Newton steps enjoy quadratic convergence near $x^\star$, but each iteration requires solving a linear system (often via Cholesky or conjugate gradients). Damped Newton with line search ensures global convergence for convex f.
- **Quasi-Newton methods.** BFGS or L-BFGS approximate Hessians by low-rank updates, offering superlinear convergence without storing the full Hessian. Widely used for logistic regression and other smooth convex losses.
- **Interior-point methods (IPM).** For constrained problems, IPMs solve a sequence of barrier subproblems
$$\phi_\mu(x) = f_0(x) - \mu \sum_{i=1}^{m} \log(-f_i(x))$$
using Newton steps. As $\mu \to 0$, solutions trace the *central path* toward the primal optimum while satisfying constraints. IPMs bring polynomial-time worst-case guarantees for LPs, QPs, SOCPs, and SDPs, albeit with higher per-iteration complexity than first-order methods.

8.6 Proximal, splitting, and decomposition techniques

When $f = g + h$ with g smooth and h simple but possibly nonsmooth, **proximal methods** exploit the proximal operator
$$\operatorname{prox}_{\eta h}(v) = \arg\min_x \left\{ h(x) + \frac{1}{2\eta} \|x - v\|^2 \right\}.$$
Common examples include soft-thresholding (for ℓ_1 norms) or projection onto convex sets. Algorithms include:

- **Proximal gradient / ISTA**: $x_{k+1} = \operatorname{prox}_{\eta h}(x_k - \eta \nabla g(x_k))$. FISTA accelerates this to $O(1/k^2)$ convergence.
- **Alternating direction method of multipliers (ADMM)**: solves $\min_x g(x) + h(z)$ subject to $Mx + Nz = c$ by alternating minimization with dual updates. ADMM decomposes problems across blocks, enabling distributed or parallel implementations (e.g., consensus optimization).
- **Augmented Lagrangian methods**: strengthen dual ascent by penalizing constraint violation quadratically, improving stability compared to vanilla dual methods.

8.7 Sensitivity, certificates, and modeling tips

- **Dual variables as sensitivities.** At optimum, $\lambda_i^\star$ equals the marginal increase in the optimal value if constraint i is tightened. Inspecting $\lambda^\star$ highlights bottleneck constraints or active data points.
- **Certificates of optimality.** A feasible primal-dual pair satisfying KKT provides a certificate. Numerically, interior-point solvers report the primal-dual gap $\|f(x) - g(\lambda, \nu)\|$, which bounds suboptimality.
- **Scaling.** Normalize features and constraints so that Hessians and gradients are well-conditioned; this accelerates both first- and second-order methods.
- **Modeling systems.** Packages like CVX, CVXPY, or disciplined convex programming (DCP) frameworks enforce convexity by construction, automatically converting high-level specifications into standard cone programs.
- **Stopping criteria.** Monitor residuals, objective gaps, and constraint violation simultaneously; mere gradient norm may be insufficient when constraints are present.

Convex optimization thus supplies the theoretical backbone for many machine learning estimators and algorithms: it furnishes tractable objectives, certifiable solutions, and scalable methods tailored to problem structure. Understanding duality and algorithmic trade-offs lets practitioners choose the right solver for each model in the AML toolbox.

9 From Convex Optimization to SVM

For support vector machines (SVMs), we're interested in maximising the margin between classes. So given a dataset $D = \{(x_i, y_i)\}$ with $y_i \in \{-1, +1\}$ and guaranteed linear separability, we want to find a hyperplane defined by (w, w_0) such that $w^T x_i + w_0 > 0$ if $y_i = 1$ and $w^T x_i + w_0 < 0$ if $y_i = -1$. Or in other terms, we want to satisfy the constraints $y_i(w^T x_i + w_0) > 0$ for all i. The margin is defined as the distance from the hyperplane to the closest data point, which can be expressed as $\frac{2}{\|w\|}$.

Derivation. Given the hyperplane $\{x \mid w^\top x + w_0 = 0\}$, the signed distance of any point x to the hyperplane is

$$\text{dist}(x, \mathcal{H}) = \frac{w^\top x + w_0}{\|w\|}.$$

Because $y_i \in \{-1, +1\}$, we can enforce the scale of (w, w_0) by requiring that the closest points satisfy $y_i(w^\top x_i + w_0) = 1$. Under this normalization, there exist parallel "margin" hyperplanes

$$w^\top x + w_0 = 1 \quad \text{and} \quad w^\top x + w_0 = -1$$

touching the positive and negative classes, respectively. The perpendicular distance between these two planes is the geometric margin:

$$\gamma = \frac{1 - (-1)}{\|w\|} = \frac{2}{\|w\|}.$$

Maximizing γ therefore amounts to minimizing $\|w\|$ (or equivalently $\frac{1}{2}\|w\|^2$ for convenience) under the constraints $y_i(w^\top x_i + w_0) \geq 1$ for all i. This yields the hard-margin SVM formulation

$$\min_{w, w_0} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^\top x_i + w_0) \geq 1,\ i = 1, \ldots, n,$$

whose solution maximizes the separating margin.

9.1 Slater's Condition

By assumption that the data is linearly separable, there exists a feasible point (w, w_0) satisfying $y_i(w^\top x_i + w_0) > 1$ for all i. This strictly feasible point ensures that Slater's condition holds, guaranteeing strong duality between the primal and dual SVM problems.

9.2 The Dual

Writing the hard-margin primal in standard form,

$$\begin{aligned} \min_{w, w_0} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(w^\top x_i + w_0) - 1 \geq 0, \quad i = 1, \ldots, n, \end{aligned}$$

introduce Lagrange multipliers $\alpha_i \geq 0$ for each margin constraint. The Lagrangian is

$$\mathcal{L}(w, w_0, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \big(y_i(w^\top x_i + w_0) - 1\big).$$

To form the dual we minimize $\mathcal{L}$ over the primal variables. Setting derivatives to zero yields the stationarity conditions

$$\nabla_w \mathcal{L} = w - \sum_i \alpha_i y_i x_i = 0 \quad \Rightarrow \quad w = \sum_i \alpha_i y_i x_i,$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = -\sum_i \alpha_i y_i = 0.$$

Substituting back gives the dual objective

$$g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^\top x_j = \sum_{i=1} \alpha_i - \frac{1}{2} ||w(\alpha)||^2,$$

with constraints $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$. Hence the dual problem is the quadratic program

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^n} \quad & \sum_{i=1} \alpha_i - \frac{1}{2} ||w(\alpha)||^2 \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, n, \\ & \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

This dual depends only on inner products $x_i^\top x_j$, enabling the kernel trick by replacing them with $k(x_i, x_j)$. The optimal weights follow from the KKT conditions: only training points with $\alpha_i^\star > 0$ (the *support vectors*) contribute to $w^\star = \sum_i \alpha_i^\star y_i x_i$. Complementary slackness enforces $y_i(w^{\star\top} x_i + w_0^\star) = 1$ for active support vectors, which can be used to recover $w_0^\star$ by averaging over any $\alpha_i^\star > 0$.

Note. The dual formulation is preferable when n (number of samples) is smaller than d (feature dimension) or when kernels project data into high-dimensional spaces. It also highlights that margin maximization depends only on a sparse subset of training examples.